CMPSC 465 Spring 2024

Data Structures & Algorithms
Mehrdad Mahdavi and David Koslicki

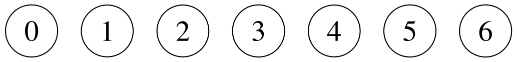
HW 8

1. (20 pts.) **Kruskal's Algorithm.** The order of edges that are added to the MST is as below with the current set(s):

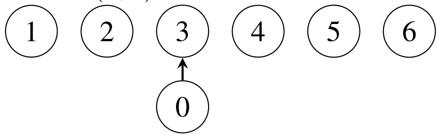
1. (a,b); current sets: $\{a,b\},\{c\},\{d\},\{e\},\{f\}$
2. (b,e); current sets: $\{a,b,e\},\{c\},\{d\},\{f\}$
3. (d,e); current sets: $\{a,b,d,e\},\{c\},\{f\}$
4. (e,f); current sets: $\{a,b,d,e,f\},\{c\}$
5. (c,e); current sets: $\{a,b,c,d,e,f\}$

2. (20 pts.) **Disjoint Set Operations.**

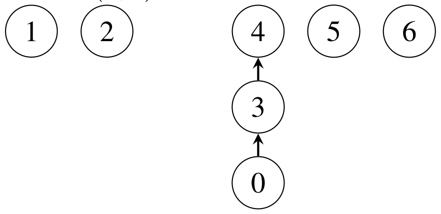
(a) Initial:



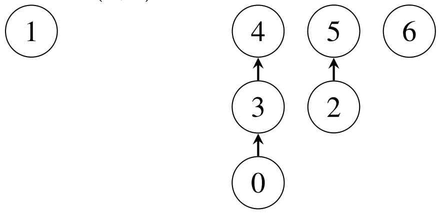
`union`$(0,3)$:



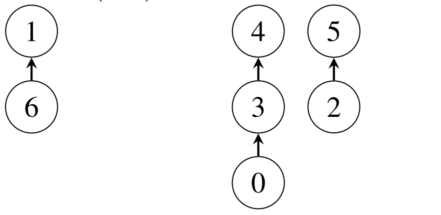
`union`$(3,4)$:



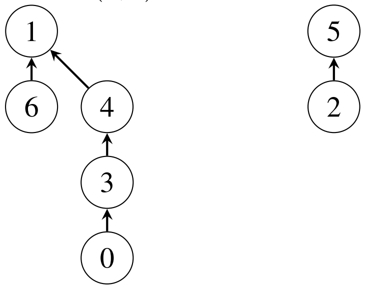
`union`$(2,5)$:



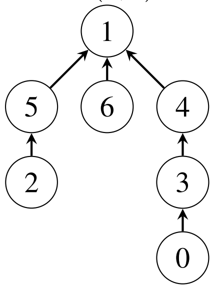
`union`$(6,1)$:



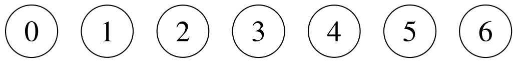
`union`$(0,6)$:



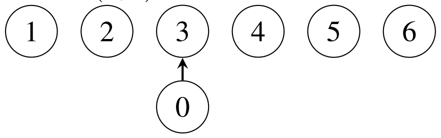
`union`$(5,1)$:



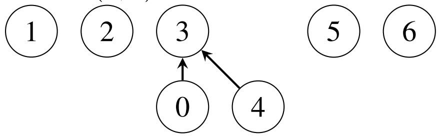
(b) Initial:



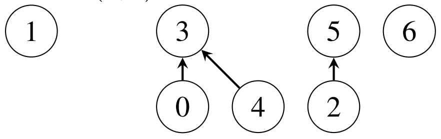
`union`$(0,3)$:



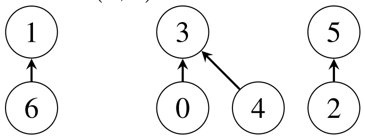
`union`$(3,4)$:



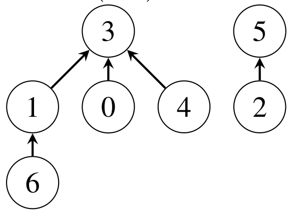
`union`$(2,5)$:



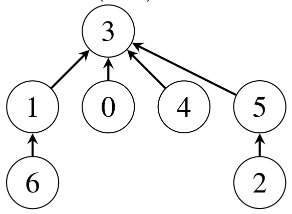
`union`$(6,1)$:



`union`(0,6):



`union`(5,1):



(c) `root` must visit each node between the desired node (here 0) and the root, inclusive. Looking at part (a)'s final tree, this requires visiting 4 of the 7 nodes, (0, 3, 4, and 1). Part (b)'s final tree requires only 2 of the 7 nodes to be visited. (0 and 3).

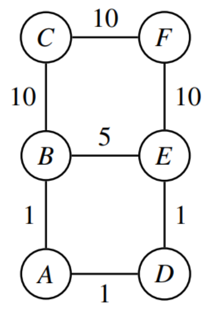
(d) The worst-case series of unions for part (a) turns the final tree into a single branch, for example, `union`(0,1), `union`(1,2), `union`(2,3), etc. Because every node is on a single path from the root, running `root` on the bottom-most node requires visiting all 7 nodes. However, the worst-case series of unions for part (b) is less clear. Because any merged tree is placed as a direct child of the root in the tree it's merging into, the height can only be increased when merging two trees of equal heights. So, creating a height 3 tree requires at least merging two height 2 trees, or 4 nodes total. Therefore, we do not have enough nodes to construct a height 4 tree, which would require at least 2 height 3 trees, or 8 nodes total. So the worst possible number of nodes on the path from a leaf to the root is 3 for part (b).

3. (20 pts.) **Maximum Spanning Tree.**

1. We make a small modification to Kruskal's algorithm. Instead of sorting all the edges based on weight in non-decreasing order, we sort them in non-increasing order for the maximum spanning tree problem. Since this only changes the sorting order, the running time of this modified Kruskal's algorithm stays the same as the original one.
2. We multiply all the edge weights with -1. Then, we run Kruskal's algorithm on this pre-processed graph. Since the pre-processing goes through all the edges, this step takes $O(|E|)$ time. Overall, this approach takes $O(|E|) + O(|E|log|V|)$ time, which is $O(|E|log|V|)$ time and the same as Kruskal's algorithm.

4. (20 pts.) **MST Basics.**

1. **True**, e will belong to the MST produced by Kruskal.
2. **True**, Suppose (u,v) is the edge. Let one side of the cut be everything reachable in the MST from u without using the edge (u,v). If this cut has an edge lighter than (u,v) then we could add this edge to the MST and remove (u,v). We know this edge is not already in the MST because otherwise both its endpoints would be reachable from u without using (u,v).
3. **False**, Let e be also the heaviest edge of a different cycle; then, we know that e can't be part of the MST. Concretely, in the following graph, edge (B,E) satisfies this condition, but will not be added to



the tree.

4. **True**, Suppose a graph G contains an r-path from a node s to t and let T be MST of G that does not contain an r-path from s to t. Then T contains a path from s to t with an edge e of weight $w_e \geq r$. Consider the partition $(S, V \backslash S)$ of vertices made by removing e from T. As r-path connects s and t there must be an edge e' of the r-path that crosses the cut $(S, V \backslash S)$. Since $w_{e'} < r$, we can swap e' for e to get a spanning tree that is lighter than T, which is a contradiction.

5. (20 pts.) **Critical edge.** Let $f > 0$ be the value of the maximum $s-t$ flow (if $f = 0$, then there are no critical edges). By the max-flow min-cut theorem the capacity of any minimum capacity $s-t$ cut is also f. Fix a minimum $s-t$ cut $\mathscr{C}$, and let e be an edge of positive capacity c_e crossing this cut. Decreasing the capacity of this edge by any $\varepsilon > 0$ decreases the capacity of $\mathscr{C}$ by ε. Since $\mathscr{C}$ was a min-cut in the original graph, the min-cut in the new graph therefore has a strictly smaller min-cut and hence a strictly smaller max-flow. Thus, any positive capacity edge which crosses an $s-t$ min-cut is a *critical edge*.

To find such an edge, we first compute the $s-t$ max flow F of value f in $G = (V, E)$, and then construct the residual graph G'. If $f = 0$, then there are no-critical edges, so we report this and exit. Otherwise, let S be the set of vertices reachable from s in G' (we know that there is no path from s to t in G', so $S \neq V$). Since there are no edges from S to $V - S$ in G', it follows that f is equal to the capacity of the cut $(S, V - S)$ in G. Thus, by the max-flow min-cut theorem, $(S, V - S)$ is a minimum cut, and from our preceding discussion, we can simply return a positive capacity edge in G that goes from S to $V - S$ (there exists such an edge since the capacity of the cut is $f > 0$).

Running time. We first do a max-flow computation. After this, constructing G' takes $O(|E| + |V|)$ time, and so does finding S (using BFS). Looking for an edge crossing $(S, V - S)$ can take a further $O(|V| + |E|)$ time, so the total running time is 1 max-flow computation $+ O(|V| + |E|)$, which is $O(|V||E|^2)$ using Edmonds-Karp.

6. (0 pts.) **Acknowledgments.**

(a) I did not work in a group.
(b) I did not consult with anyone other than my group members.
(c) I did not consult any non-class materials.

Rubric:

Problem 1, 20 pts

- 2 pts for each correct edge stated in the correct order
- 2 pts per step for the vertex sets (order of elements in each set does not matter)

Problem 2, 20 pts

(a) 6pts, 1 for each union forest.

(b) 6pts, 1 for each union forest.

(c) 4pts for the correct answer, (a) requires 4, (b) requires 2.

(d) 4pts for the correct answer, (a) requires at most 7, (b) requires at most 3.

Problem 3, 20 pts

For each of the two approaches:

7 points: provide a correct approach and describe how it works
3 points: explain why this approach has the same complexity as Kruskal's algorithm
In the case of multiplying all edges with -1 then running Kruskal's algorithm: 2 points for pointing out the pre-processing takes $O(|E|)$ time, and 1 point for concluding the overall running time is $O(|E|log|V|)$.

Problem 4, 20 pts

- 2pts for correct T/F for each part
- 3pts for correct proof/counterexample for each part

Problem 5, 20 pts

1. 10 pts for showing that edges on the minimum cut are critical edges.
2. 6 pts for using augmentation path or other method to find the minimum cut.
3. 4 pts for showing the correct running time.